A Helmholtz equation solver using unsupervised learning: Application to transcranial ultrasound

Antonio Stanziola[1,*], Simon R. Arridge[2], Ben T. Cox[1], and Bradley E. Treeby[1]

[1]Department of Medical Physics and Biomedical Engineering
University College of London, Gower Street, London WC1E 6BT, UK

[2]Department of Computer Science
University College of London, Gower Street, London WC1E 6BT, UK

[*]a.stanziola@ucl.ac.uk

June 8, 2021

Abstract

Transcranial ultrasound therapy is increasingly used for the non-invasive treatment of brain disorders. However, conventional numerical wave solvers are currently too computationally expensive to be used online during treatments to predict the acoustic field passing through the skull (e.g., to account for subject-specific dose and targeting variations). As a step towards real-time predictions, in the current work, a fast iterative solver for the heterogeneous Helmholtz equation in 2D is developed using a fully-learned optimizer. The lightweight network architecture is based on a modified UNet that includes a learned hidden state. The network is trained using a physics-based loss function and a set of idealized sound speed distributions with fully unsupervised training (no knowledge of the true solution is required). The learned optimizer shows excellent performance on the test set, and is capable of generalization well outside the training examples, including to much larger computational domains, and more complex source and sound speed distributions, for example, those derived from x-ray computed tomography images of the skull.

Keywords Helmholtz equation · learned optimizer · unsupervised learning · physics-based loss function · transcranial ultrasound

1 Introduction and background

1.1 Motivation

Transcranial ultrasound therapy is a rapidly emerging technique for the noninvasive treatment of brain disorders in which ultrasound is used to cause functional or structural changes to brain tissue. Several different types of treatment are possible depending on the pattern of ultrasound pulses used and the addition of exogenous microbubbles. This includes precisely destroying small regions of tissue [1], generating or suppressing electrical signals in the brain [2], and temporarily opening the blood-brain barrier to allow drugs to be delivered more effectively [3]. A major challenge for transcranial ultrasound therapies is the presence of the skull bone, which causes the ultrasound waves to be distorted and attenuated, even at low frequencies [4]. Critically, the skull morphology and acoustic properties vary both within and between individuals, which leads to undesirable changes in the position and intensity of the ultrasound focus [5, 6], and in some cases can destroy the focus entirely [7].

Using computational ultrasound models and knowledge of the geometric and acoustic properties of the skull (e.g., derived from an x-ray computed tomography image), it is possible to predict the ultrasound field inside the brain after propagating through the skull, and thus account for subject-specific dose and targeting variations [8, 9]. However, existing models based on conventional numerical techniques typically take tens of minutes to several hours to complete due to the large size of the computational domain compared to the size of the acoustic wavelength, in some cases generating models with billions of unknowns which require tens of thousands of iterations to solve [10–14]. This makes them too slow to be used for online calculations and corrections, i.e., while the subject is undergoing the therapy. Consequently, approximate models are often used (e.g., ray tracing) which trade off between accuracy and computational efficiency [15, 16].

In the current work, instead of using a classical numerical partial differential equation (PDE) solver, a recurrent neural network architecture is developed to rapidly solve the Helmholtz equation with a heterogeneous sound speed distribution representative of a human skull. The network is trained using a physics loss term based on the Helmholtz equation and a training set of idealized sound speed distributions. The use of a physics loss term, which plays an analogous role to the data consistency term in inverse problems [17, 18], avoids the need to run a large number of computationally expensive simulations using a conventional solver to generate training data for supervised training. A review of the relevant background to this approach is described in the remainder of §1, with the developed network architecture and training outlined in §2. Results are then given in §3 with discussion and outlook in §4.

1.2 Governing equations

In the most general case, the propagation of ultrasound waves through the skull and brain involves a heterogeneous distribution of material parameters, shear wave effects in the skull, nonlinear effects when high intensities are used, and acoustic absorption [11, 19]. However, if the ultrasound waves approach the skull close to normal incidence (this is often the case), then shear motion can be ignored [20]. Moreover, nonlinear effects are only important for ablative therapies and are restricted to a small region near the focus [21], and acoustic absorption can be considered a second-order effect [22]. In addition, for many therapeutic applications, the applied ultrasound signals are at a single frequency and last for many milliseconds or seconds, which is typically much longer than the time taken for the acoustic field to reach a steady-state, and thus time-independent models can be used.

Considering the above, a simplified model of wave propagation through the skull and brain can be described by the heterogeneous Helmholtz equation subject to the Sommerfeld radiation condition at infinity:

$$\left[\nabla^2 + \left(\frac{\omega}{c(r)} \right)^2 \right] u(r) = \rho(r) \ , \tag{1}$$

$$\text{s.t.} \lim_{|r| \to \infty} |r|^{\frac{n-1}{2}} \left(\frac{\partial}{\partial |r|} - i \frac{\omega}{c_0} \right) u(r) = 0 \ . \tag{2}$$

Here n is the number of spatial dimensions, $c : \mathbb{R}^n \to \mathbb{R}^+$ is the speed of sound, ω is the angular frequency of the source, $r \in \mathbb{R}^n$ is a general space coordinate, $\rho : \mathbb{R}^n \to \mathbb{C}$ is the source distribution, and $u(r) \in \mathbb{C}$ is the complex acoustic wavefield. Here, it is assumed that the speed of sound distribution $c(r)$ is heterogeneous in a bounded region of the domain, while it is uniform and equal to c_0 outside of it. In practice, the solution to Eq. (1) is sought within a domain of interest $\Omega \subset \mathbb{R}^n$ as shown in Fig. (1).

The aim of the current work is to find a learned iterative solver capable of generating a field u which satisfies Eqs. (1) and (2). We will focus on the 2D version of the Helmholtz problem, keeping in mind that the ultimate goal is to translate our findings to large three-dimensional simulations. Extensions to 3D and to other wave equations, e.g., to include the effects of nonlinearity, changes in mass density, and acoustic absorption, will be considered as part of future work.

1.3 Numerical solution techniques

The Helmholtz equation given in Eq. (1) can be written in the general form

$$A(c)u = \rho \ , \tag{3}$$

where $A(c)$ is a linear forward operator which depends on the speed of sound distribution c. There are many approaches to discretize A, including finite difference methods [23], boundary element methods [24], and finite-element methods [25]. In many cases, direct inversion of the forward operator is not feasible due to the
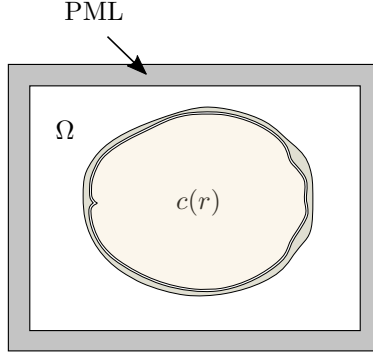
Figure 1: Definition of the computational domain Ω which contains a heterogeneous sound speed $c(r)$, in this case represented by a skull. A perfectly matched layer (PML) is used to surround the computational domain to simulate exterior open boundaries as discussed in Appendix A.

size and conditioning of the problem, and thus iterative schemes are used. In this case, the solution of the PDE is cast as an optimization problem via a suitable loss function, which is solved using a minimization algorithm.

The most widely used loss L is the squared norm of the residual e_k, which is equivalent to the mean squared error (MSE) up to a scaling factor. Given the solution u_k at iteration k over the domain of interest Ω, this can be calculated by [26]:

$$L_k(u_k, c, \rho) = \|e_k\|^2 = \int_\Omega |e_k|^2 dr \;, \qquad e_k = A(c)u_k - \rho \;. \tag{4}$$

Other objective functions may also be used depending on the required characteristics of the solution or the discretization model at hand. Common alternative choices are the root-MSE (RMSE) or the mean absolute error (MAE). Other examples relevant to the solution of PDEs include using the Dirichlet energy [27], augmenting the MSE loss with an ℓ_∞ term to enforce strong PDE solutions [28], and using physics-constrained loss functions [29].

Given a suitable discretization and loss function, a common approach to solving the system of equations is the use of Krylov subspace methods, such as the widely used conjugate gradient (CG) and generalized minimal residual (GMRES) methods [30]. However, Krylov subspace methods are known to have a slow convergence rate for the Helmholtz problem [31]. From an intuitive perspective, if the solution starts with an empty wavefield and a spatially localized source, the nature of the Helmholtz equation makes each update of Krylov methods local (due to the Laplacian).[1] This means the solution will grow slowly from the source position (an example is shown later in the results section in Fig. 9). However, the solution to the heterogeneous Helmholtz problem clearly has non-local dependencies, for example, a strong reflector at one end of the domain will affect the wavefield at the other end.

To mitigate these issues, preconditioning methods can be used. Mathematically, this means finding a suitable change of basis that reduces the dynamic range of the singular values of the forward operator, which often leads to the use of a multiscale representation that can take into account long-range dependencies [32, 33]. However, finding suitable preconditioning methods for wave problems is a challenging task. Algebraic methods, such as sweeping preconditioners, are often based on matrix factorization methods that require instantiating the numerical discretization of the operator [34], which is often prohibitive for spectral methods that implicitly assume dense matrices.

Standard approaches to adapt multigrid preconditioners for the Helmoltz equation require careful rethinking of the standard components of multigrid methods, especially for coarse grids. This makes off the shelf multigrid methods, such as those based on damped Jacobi smoothing, effective only for low wavenumbers [35]. One of the first successful attempts is given in [35], where the authors used GMRES as a smoothing operator to effectively suppress high-frequency components. While this produced impressive convergence results for large-wavenumber problems, each iteration is computationally demanding due to the internal iterative solvers

---

[1]Krylov methods work by building a sequence of basis vectors $v_{j+1} = A^{\mathrm{T}} A v_j$ and finding the optimal solution in the subspace spanned by this sequence up to the current value of j. Thus the spatial support of the solution can only increase by the support of $A^{\mathrm{T}} A$ on each iteration, which is local for differential operators.

applied before each restriction operation. This idea was extended in [36] by learning the optimal subspace for GMRES, instead of relying on the classical Krylov subspace, although results are only shown for parabolic PDEs.

Other authors proposed to use a shifted-Laplacian preconditioner which can be inverted using multigrid methods [32]. While effective, the multigrid iteration used for preconditioner inversion is still limited in its deepest coarse correction by the eigenvalue structure of the preconditioner itself, which is close to the original Helmholtz operator. Furthermore, convergence analysis and computation of the preconditioner's inverse is in most cases performed with a finite-difference discretization [33], often relying on the sparse structure of the discretized operator to perform efficient computations, which is not applicable to spectral discretizations that produce much denser matrices. The lack of a sparse structure is also a challenge for incomplete LU decomposition techniques.

An alternative idea towards improving Krylov-based iterative schemes is given in [37], where Krylov iterations are interleaved with a UNet [38] that *boosts* the current solution towards the true one. In [37], the UNet is trained on known examples, however, training on unlabeled examples using a physics loss is also suggested. A nice advantage of keeping some Krylov iterations is that they may act as regularizers, preventing the solution from diverging.

More generally, this leads to the following question: instead of using traditional optimizers to solve Eq. (3), is it possible to *learn* a suitable optimizer f that can be iteratively applied to generate an accurate solution in a small number of iterations, while at the same time being simple enough to be used on large scale problems? While the general idea of learning an optimizer is not new [39, 40], fully learned optimizers for the Helmholtz equation have not previously been explored, and the task involves a problem-specific trade-off between speed (number of iterations) and accuracy (how close we are to the solution).

1.4 Learned optimizers

In the case of the heterogenous Helmholtz equation given in Eq. (1), assuming a fixed angular frequency ω, an iterative optimization scheme could be written in the form

$$u_{k+1} = u_k + f_\theta(u_k, c, \rho), \tag{5}$$

where f_θ is a learned function parametrised by the estimate of the wavefield u_k after the kth iteration, the sound speed distribution c (which is used in the forward operator A), and the source distribution ρ. However, a key problem with this formulation is that it is hard for the network to directly manipulate the sound speed and source distributions to give an update for u, as they belong to two very different domains. Although some learning-based methods have been proposed to combine samples from different domains, such as AUTOMAP [41], it is hard to design a function approximator with the right inductive biases while preserving the necessary flexibility required for fast inference. This often means that fully connected deep networks must be used, which are hard to scale to high dimensional input/output pairs and require a large amount of data to be trained.

Instead of using Eq. (5), we can instead leverage our knowledge of the forward operator to manipulate c and ρ to get a new input e_k, which belongs to the same domain as u_k. For example, we could use the residual signal e_k as an input

$$u_{k+1} = u_k + f_\theta(u_k, e_k) \ , \qquad e_k = A(c)u_k - \rho \ . \tag{6}$$

(Other choices are also possible for e_k, for example, the derivative of the loss function.) This approach is in direct parallel to the inputs to many optimization algorithms. However, Eq. (6) still assumes that the combination of the current wavefield estimate and latest residual gives enough information to the network to specify the next update. This is unlikely to be true, as for different problems, the same residual could be observed given the same wavefield. A simple example is for an empty wavefield and a fixed source distribution ρ, the residual will be the same regardless of the sound speed distribution. If we look at the choice of the sequence of updates as a Markov decision process (MDP) [42], this makes it only partially observable.

One way to restore the properties of a fully observable MDP is to construct a recurrent *belief state* h which contains information from all preceding actions and observations [43]. Augmenting the problem with such a state can be done via the following update rule:

$$\begin{aligned} (\Delta u_{k+1}, h_{k+1}) &= f_\theta(u_k, e_k, h_k) \\ u_{k+1} &= u_k + \Delta u_{k+1} \ . \end{aligned} \tag{7}$$
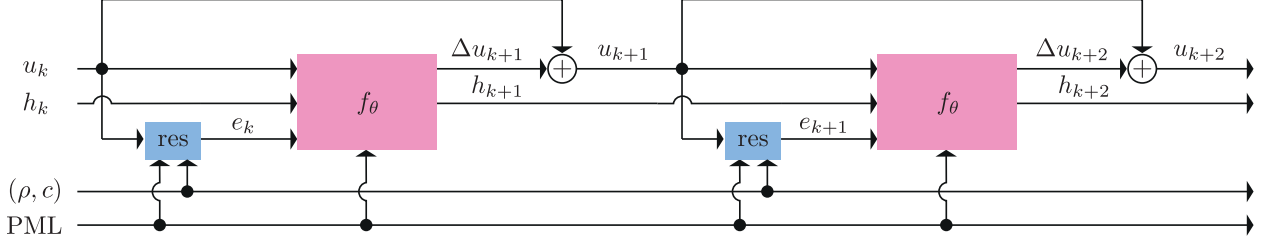
Figure 2: Schematic of the proposed iterative scheme (showing two iterations) for the solution of the heterogeneous Helmholtz equation u. This uses a fully-learned optimizer f_θ which outputs both the solution update Δu_k along with a belief state h_k. The residual e_k is calculated using a physics loss (represented in the figure by the block res). This is different for every speed of sound distribution and is used as an additional input to the network. Both the optimizer and the residual calculation make use of knowledge of the perfectly matched layer (PML). The residual calculation also uses the sound speed c and source distribution ρ through the forward operator.

Note that the wavefield update resembles a discrete Euler stepping scheme [44]. Augmenting the input with a hidden state is known to improve the representation capabilities of neural ODEs [45] and it is therefore reasonable to assume it also helps in their discretized counterpart.

If the function f_θ is considered as an iterative solver, the presence of the state variable h allows several optimizers to be cast in this framework. For example, if h stores the previous gradient and its magnitude, f_θ could in principle work as a quasi-Newton method [39, 40], while if it stores the collection of all the previous residuals, it may generalize Krylov subspace methods.

A learned iterative solver in this framework was proposed by Putzky and Welling [40] for various kinds of image restoration tasks in which e_k was given by the gradient of the loss function (in their case, the log likelihood for a Gaussian prior). This scheme was later extended by Adler and Oktem [46] for non-linear operators and applied to a non-linear tomographic inversion problem.

2 Network architecture and training

2.1 Iterative solution to the Helmholtz equation

Building on the work discussed in §1.4, we propose an iterative method in the form of Eq. (7) to solve the Helmholtz equation using a learned optimizer. The discrete dynamical system that models the iterative solution u_k of the heterogeneous Helmholtz equation given in Eq. (1) can be written as

$$
\begin{aligned}
e_k &= \left[\nabla^2 + \left(\frac{\omega}{c} \right)^2 \right] u_k - \rho \\
(\Delta u_{k+1}, h_{k+1}) &= f_\theta(u_k, e_k, h_k) \\
u_{k+1} &= u_k + \Delta u_{k+1} \quad .
\end{aligned}
\tag{8}
$$

Here f_θ is the neural network with learnable parameters θ, e_k is the residual computed from the heterogeneous Helmholtz equation, Δu_k is the iterative update, and h_k is a learned hidden or belief state. A diagram showing two unrolled iterations is given in Fig. 2. The network also uses an auxiliary input of the variable absorption coefficients that characterize the perfectly matched layer (PML) in each Cartesian direction (the PML is used as part of the discretized Laplacian operator to mimic the radiation condition in Eq. (2) as outlined in Appendix A). This allows the network to learn how to dampen the waves at the edges of the domain to simulate exterior open boundaries, and to appropriately weight the corresponding residuals.

The network is trained using a physics-based loss function. This avoids the need for labeled training data, e.g. created by running a large number of simulations using a conventional PDE solver to obtain ground truth solutions. While generating sufficient labeled training data may technically be feasible in 2D, for 3D problems, running a single simulation can take tens of minutes to many hours [11], which makes the generation of a large training set practically intractable.

If the total number of iterations is fixed to T (i.e., a finite horizon), one approach for training the network would be to minimize L_T, the final loss calculated after performing T iterations. However, in practice, it is
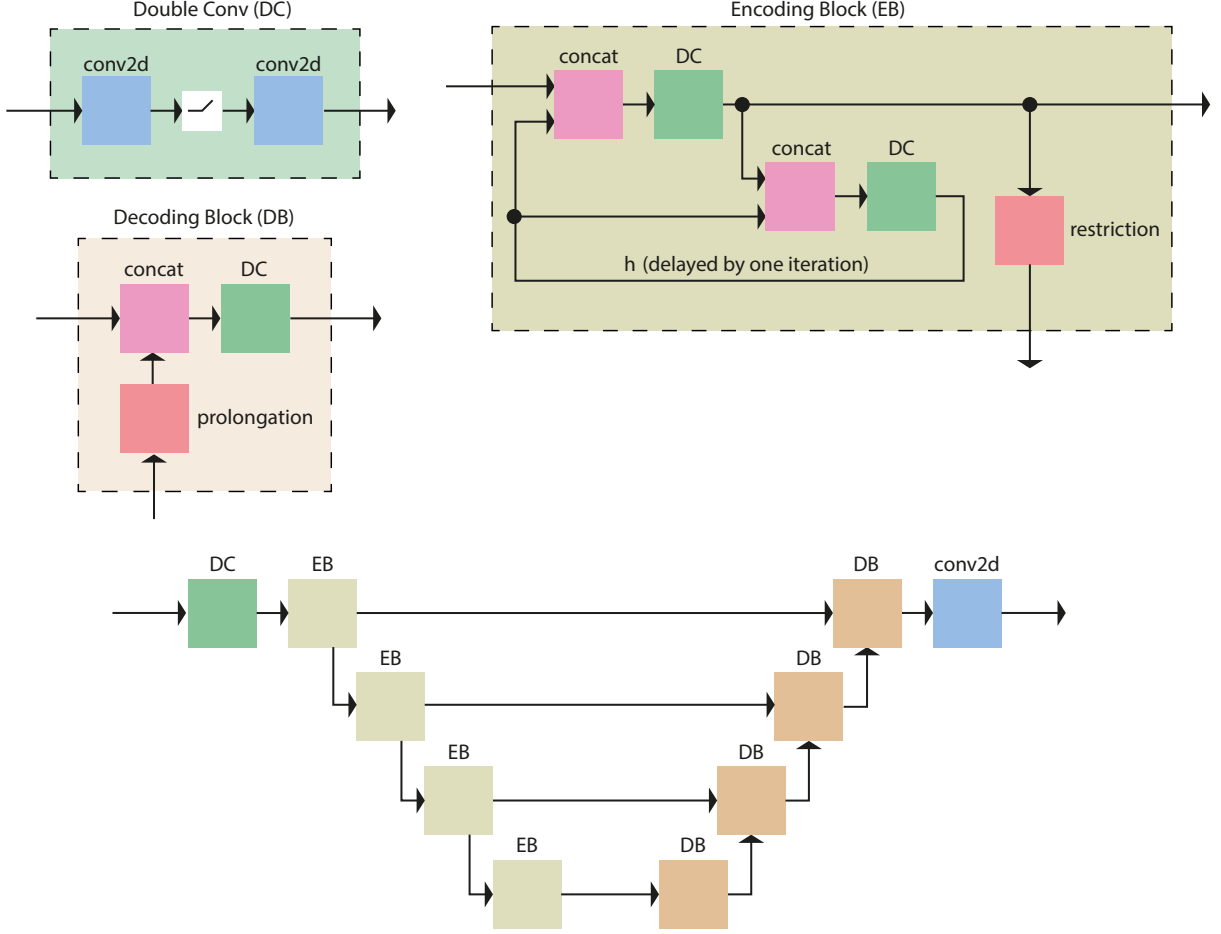
Figure 3: Architecture of the modified UNet used for the learned optimizer f_θ. Each encoding block (EB) contains two double convolution (DC) layers, one to compute the output passed to subsequent layers, and one to compute the hidden state h. The concat blocks stack the inputs in the channel dimension. The network is lightweight, with only 8 channels per convolutional block at every scale and a total of 47k trainable parameters.

not computationally feasible to perform backpropagation through a large number of iterative steps of the learned optimizer. The optimal choice of T is also not straightforward, as the accuracy required and the number of iterations to reach that accuracy are problem specific.

Instead, we choose to minimize the total loss function R across all iterations, which is given by

$$R = \sum_{k=0}^{T} w_k L_k(u_k, c, \rho) \ , \tag{9}$$

where L_k is the physics-based loss function calculated after each iteration, and $w_k \geq 0$ are a set of weights that define the importance of the loss at each step. Here, the loss from Eq. (4) is used with the residual from Eq. (8). The weights are set to $w_k = 1 \forall k$, in other words, we aim to minimize the loss at every step of the iteration, on average, which yields

$$R = \sum_{k=0}^{T} \|e_k\|^2 \ . \tag{10}$$

To avoid unrolling the network for a large number of iterations for backpropagation, a replay buffer is used as discussed in detail in §2.3.

2.2 Neural network architecture

The neural network architecture for the optimizer f_θ is depicted in Fig. 3. The input to the network has the same spatial dimensions as the sound speed distribution (which is defined on a regularly spaced Cartesian grid) and contains six channels: two channels for the real and imaginary parts of the wavefield u_k, two channels for the real and imaginary parts of the residual e_k, and two channels for the variable absorption coefficients used in the PML in each Cartesian direction ($\sigma(x)$ and $\sigma(y)$ defined in Appendix A).

The core building block of the network is the widely used double-convolution layer, which consists of two bidimensional convolutions with 3-by-3 kernels, interleaved by a parametrized-ReLU non-linear activation function [47]. Each encoding block contains two double-convolution layers. The first accepts two inputs: the network input representation at the current scale and a hidden state. The output is then passed to a second double-convolution layer (used to update the hidden state), the corresponding decoding block of the network, and a restriction operator which downsamples the output and feeds it to a deeper layer of the network. The restriction operator is represented by an 8-by-8 convolutional stage, applied with stride 2 in order to halve the dimension of the wavefield at each depth, reaching a total of 4 depths.

Internally, each encoding block stores its own hidden state, which from a functional point of view can be considered as being passed between iterations, as shown in Fig. 2. The size of the hidden state for each encoding block matches the size of its corresponding input, and has two channels. Note, the state variable h_k in Eq. (8) and Fig. 2 refers to all the states stored across all encoding blocks.

The decoding blocks take an input from the layer below, upsamples it using transposed convolutions with 8-by-8 kernels and stride 2, and after concatenating it with the output from the corresponding encoding block (i.e., the skip connection), produces an output via another double-convolution layer. Finally, the last layer of the network is a 1-by-1 convolution that maps the output of the neural network to the wavefield domain. The output has the same spatial dimensions as the input and contains two channels for the real and imaginary parts of the wavefield.

There are several intuitions behind this choice of architecture. First, having a fully convolutional network implicitly imposes some degree of translation invariance to the iterative solver,[2] while at the same time allows the network to be used with arbitrarily-sized sound speed distributions. Second, the network can encode priors at different scales thanks to the multiscale structure, allowing correction for very local distortions of the wavefield while at the same time taking care of long range dependencies. This is very similar to the idea behind the MultiGrid Neural Network (MgNet) [48], which connects UNet-like architectures with the theory of multiscale solvers, the latter widely used to solve the Helmholtz problem.

In total, the network has approximately 47k trainable parameters, with only 8 channels per convolutional block at every scale. The very small network size is possible because the solution is iteratively updated using the residuals of the true forward operator. [3]

2.3 Training

The neural network is trained on a dataset of sound speed distributions containing idealized skulls. All calculations are performed in normalized units with a source frequency of $\omega = 1$ rad/s and a background sound speed of 1 m/s. The idealized skulls are randomly generated with a hollow convex structure with a constant thickness and constant speed of sound, defined by summing up several circular harmonics of random amplitude and phase. Between examples, the skull thickness ranges from 2 to 10 m and the sound speed from 1.5 to 2 m/s giving a maximum sound speed contrast of 100 % (this matches the sound speed contrast between soft tissue and human skull bone [49]). The size of each example is 96×96 grid points with a normalized grid spacing of 1 m, giving 2π points per acoustic wavelength (PPW). Note that while the training is performed using normalized units, the results can be re-scaled for any combination of grid spacing / source frequency / background sound speed, provided the PPW remains fixed at 2π (an example of simulating a transcranial ultrasound field within an adult skull at 490 kHz is discussed in Sec. 3.3).

The training set contains 9000 sound speed distributions while both the validation and test sets contain 1000 distributions. Several examples are shown in Fig. 4. As a physics-based loss function is used, the training data only contains sound speed distributions—no ground truth wavefields (e.g., generated using another PDE

---

[2]For example, if ρ and c were shifted upwards by an equal amount, we would expect the solution also to be shifted upwards by the same amount, i.e., for it to be translationally invariant. Therefore it is desirable for the neural network architecture to also be translationally invariant.

[3]Code will be made publically available on GitHub after publication of the paper
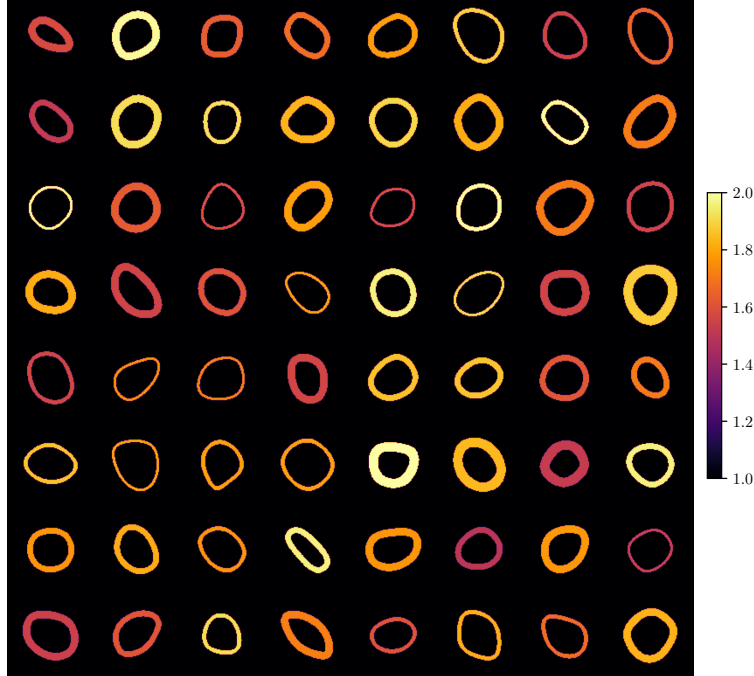
Figure 4: Examples of the sound speed distributions based on idealized skulls used to train the learned optimizer. Each skull is created by summing up several circular harmonics of random amplitude and phase, and then assigned a random thickness between 2 and 10 pixels, and a random sound speed between 1.5 and 2 times the background value.

solver) are required. During training, the source distribution ρ is always fixed to a single grid point with magnitude 10 at the bottom of the domain.

From a practical point of view, it is not computationally feasible to perform backpropagation through a large number of iterative steps of the learned optimizer. To overcome this, the network is trained using a replay buffer and truncated backpropagation through time (TBPTT) [50], where TBPTT is implemented by unrolling 10 iterations. The replay buffer is initially filled with 600 triplets (c, u_k, h_k) containing sound speed distributions randomly selected from the training set. For each sound speed example, the wavefield, and hidden state are initialized to zero, while the iteration index k is initialized as a random integer between 0 and the maximum number of iterations, in this case set to $T = 1000$.

During training, at each training step a mini-batch of N triplets (containing examples with a range of different sound speed distributions and iteration indices) is randomly selected from the buffer. For each triplet, the loss is calculated over 10 iterative steps using Eq. (4), and the total loss is then summed over the mini-batch, where

$$R = \frac{1}{N} \sum_n \sum_k L_k(u_{k,n}, c_n, \rho) \ . \tag{11}$$

Here $n \in \{0, \ldots, N - 1\}$ is the triplet index over the mini-batch, $k \in \{k_n + 1, \ldots, k_n + 10\}$ is the iteration index, and k_n is the starting iteration index for the n-th triplet. Finally a gradient descent step is performed over 10 unrolled iterations to update the network.

The calculation of the loss L_k using Eq. (11) is performed using a Fourier collocation spectral method with a modified Laplacian that includes a PML. This allows the Sommerfeld radiation condition at infinity to be approximately satisfied while cropping the domain to a finite size $\Omega \subset \mathbb{R}^2$. The discrete formulation used is given in Appendix A.

After each training step, for each example in the mini-batch, one of the iterative steps is randomly selected, and a new triplet (c, u_k, h_k) is stored back into the buffer replacing the previously selected triplet. For computational reasons, the residual is also stored alongside the triplet to avoid needing to recalculate this

when the triplet is later re-drawn from the buffer. During training, if the iteration index of any of the triplets to be stored back into the buffer exceeds $T = 1000$, this is replaced with a new sound speed distribution randomly selected from the training set. The wavefield, hidden state, and iteration index for the new example are initialized to zero. The full training algorithm is given in Appendix B.

Note, the use of a small window for backpropagation can bias the network towards learning state representations which have short temporal dependency. While there are techniques for mitigating such bias [51], we didn't find this to be a problem in practice. Storing the belief state h_k in the replay buffer (rather than re-initializing it) has also been shown to improve the performance of recurrent networks trained using experience replay [52].

Gradient descent is performed using Adam with a batch size of 32, learning rate of 10^{-4}, and gradient clipping at 1. The biases of all convolutional layers are initialized to zero to minimize the risk of divergence of the wavefield in the early iterations. We also don't store triplets in the buffer if the loss goes above an arbitrary threshold value of $L = 1$, as this suggests that the wavefield is diverging. We found this to be especially important in the early phase of training.

The network and training were implemented using PyTorch and parallelized using PyTorch Lightning [53]. The training was performed using a cluster of 6 NVIDIA Tesla P40 graphics processing units (GPUs) in parallel. During training, at the end of every 10 epochs, the loss on the validation set was also evaluated, in this case by summing the loss at the last iteration L_T over all examples in the validation set. However, in this case the source position ρ for each example was moved to a random position on a circle to provide a simple test of network generalization. Since no input/output pairs were provided during training, inputs and outputs were scaled by a factor of 10^3 and 10^{-3}, respectively. These values were hand-tuned to roughly normalize the variance of the inputs and outputs across all iterations. Similarly, the loss function was amplified by a factor of 10^4. The training was run for 1000 epochs (52k training steps), and the network with the lowest validation loss was selected. The total training time was approximately 21 hours. A summary of the network and training hyperparameters is given in Appendix C.

3 Results

3.1 Model accuracy against a reference solution for the test set

To evaluate the performance of the trained network, a series of tests were performed. First, the accuracy of the network for the (unseen) sound speed maps in the test set was evaluated by comparing the wavefields calculated after 1000 iterations of the learned optimizer against a reference solution. The reference solution was calculated using the open-source k-Wave acoustics toolbox [54]. To obtain a time-independent solution to the wave equation, the time-domain solver `kspaceFirstOrder2DG` was used with a continuous wave sinusoidal source term, the solution was run to steady state, and the complex wavefield extracted by temporal Fourier transform. The wavefields were then normalized to an amplitude of 1 and phase of zero at the source location to account for the different source scaling and relative phase between the two models. The accuracy was computed using the relative ℓ_∞ and average RMSE error norms calculated as

$$\ell_\infty = \frac{\|u_{\text{predicted}} - u_{\text{reference}}\|_\infty}{\|u_{\text{reference}}\|_\infty} \quad , \qquad \text{RMSE} = \sqrt{\frac{\|u_{\text{predicted}} - u_{\text{reference}}\|_2^2}{N}} \quad , \tag{12}$$

where N is the total number of pixels in the wavefield. As the learned optimizer and k-Wave use different formulations for the PML, this region was excluded from the error calculations.

A histogram of the error norms for the 1000 examples in the test set is shown in Fig. 5, with four examples of the calculated wavefields given in Fig. 6. The predicted wavefields have very low errors compared to the reference solution, with a mean ℓ_∞ error of 0.36%, and mean RMSE of 4.6×10^{-4}. This demonstrates that the learned optimizer gives highly accurate results. Although not a focus of the current work, preliminary benchmarks show the learned model to be at least an order of magnitude faster than k-Wave for the same level of accuracy.

During the iterative procedure, it is possible to monitor the progression of the solution using metrics based on the computed residual. Figure 6 gives four examples of the evolution of the residual with iteration number, and Fig. 8 for all examples in the test set. Typically, a few hundred iterations are needed for the residual magnitude to reach a minimum. However, while a zero residual indicates convergence to the true solution, it is not immediately obvious how other values correspond to absolute accuracy. To investigate this, the evolution of the residual magnitude vs ℓ_∞ error with iteration number for the test set is plotted in Fig. 7. In general, the curves decrease with iteration number, meaning a lower residual magnitude gives a lower ℓ_∞
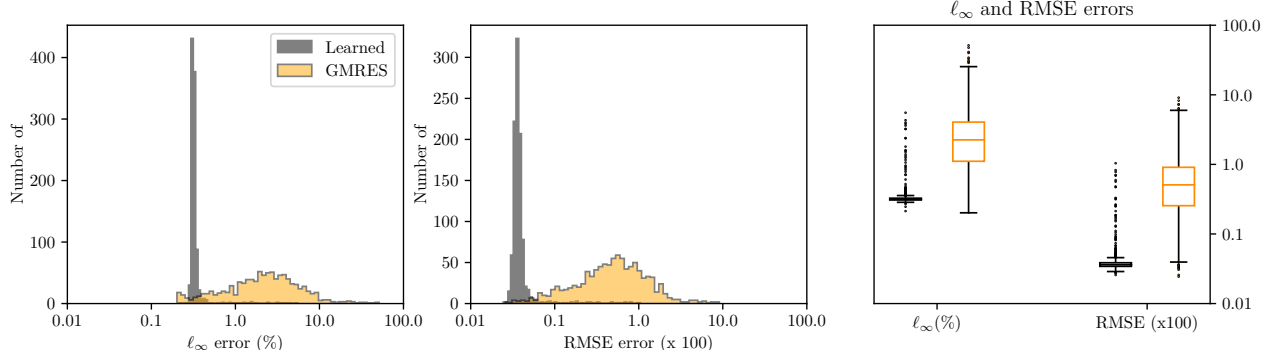
Figure 5: Errors in the wavefields predicted by the learned optimizer and the generalized minimal residual (GMRES) method shown as histograms (left and middle) and box plots (right). The errors are compared after 1000 iterations against a reference solution calculated using k-Wave for the 1000 sound speed distributions in the test set. The plotted RMSE errors are multiplied by 100 to move them to a similar scale. The learned optimizer is highly accurate, with a mean ℓ_∞ error of 0.36%, and mean root mean square error (RMSE) of 4.6×10^{-4}.

error for a given problem. However, while a high residual magnitude (e.g., 10^{-3}) implies a high error, and a very low residual (e.g., 2×10^{-5}) implies a low error, for intermediate values (e.g., 10^{-4}), there is significant spread. This will be investigated further in future work.

In both Fig. 5 (right panel) and Fig. 7, a small number of outliers can be seen with errors on the order of a few percent. In general, these outliers had a higher value for the final residual, and the evolution of the residual often displayed oscillations. Further investigation into these examples demonstrated the presence of a mode-like structure in the wavefield within and adjacent to the idealized skull, similar to whispering gallery modes. These examples were also much more difficult for k-Wave to compute, requiring at least twice as many time steps to reach an approximately steady state. Empirically it seems that examples in which highly resonant modes are supported take longer to converge, which suggests there may be a connection between the iterations and the time evolution of the field, as shown in Figs. 9 and 13. The discrepancy between the mean and median error in the early steps of optimization as shown in Fig. 8 also suggests that there are some particular sound speed distributions in the test set which are more challenging for the learned optimizer to solve. Interestingly, the same behavior was not observed for more complex sound speed distributions, e.g., based on real skulls.

3.2 Comparison with GMRES

To benchmark against classical methods, we compared the learned iterative solver with the widely used GMRES method for the sound speed examples in the test set. Figure 5 shows histograms and box plots of the ℓ_∞ and RMSE errors against k-Wave, while Fig. 8 shows the progression of the residual and ℓ_∞ against iteration number. The learned optimizer outperforms the generic solver both in terms of convergence speed with iteration number, and in terms of accuracy for a given number of iterations. After 200 iterations, the learned optimizer reaches an average residual magnitude of about 5.6×10^{-5} and ℓ_∞ error of 1.0%. In comparison, GMRES reaches an average residual magnitude of about 5.7×10^{-3} and ℓ_∞ error of 20.4%. Even after 1000 iterations (a five-fold increase), GMRES only reaches an average ℓ_∞ error of 2.8%.

The difference in the convergence rates can be understood by looking at the evolution of the solution with iteration number for a representative example as shown in Fig. 9. While both GMRES and the proposed solver construct the solution from the source location outwards, the spatial extent of the update made at each step by the two are very different: while GMRES tends to make very local updates (this is due to the nature of the Krylov iterations using a local forward operator, as discussed in §1.3), the learned iterative solver updates the solution over much larger spatial ranges.

3.3 Network generalizability

Having established the ability of the learned optimizer to give highly accurate solutions for the sound speed distributions in the test set, a preliminary investigation was performed into its generalization capabilities
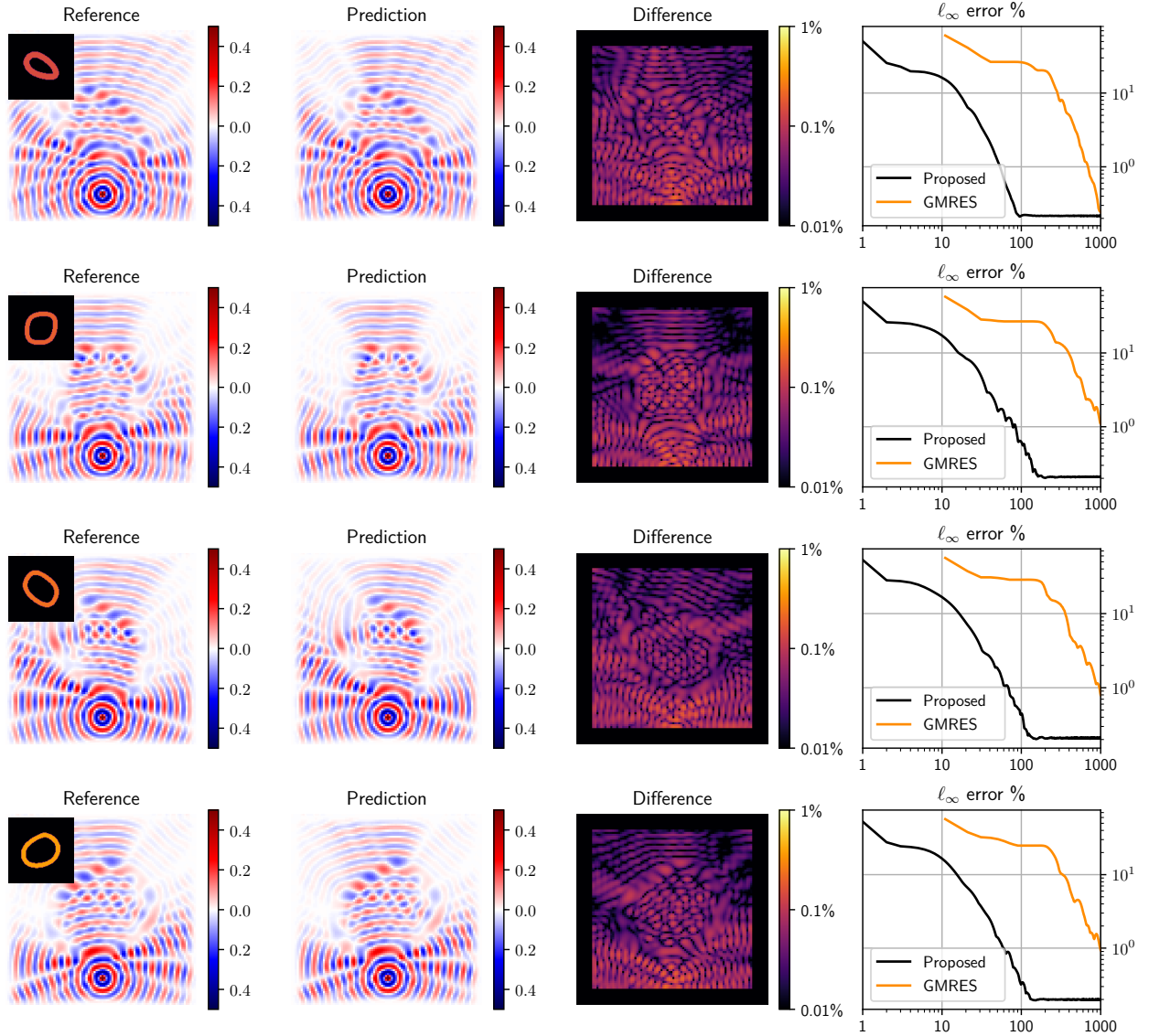
Figure 6: Four examples of simulations using idealized skull distributions randomly selected from the test set (insets shown top left). In each case, the reference solution is computed using k-Wave and shows very close agreement with the prediction using the learned optimizer (the real part of the wavefield is shown). For these examples, the ℓ_∞ error reaches a minimum within 200 iterations. The convergence of GMRES is also shown for comparison: we reset the Krylov subspace every 10 iterations.
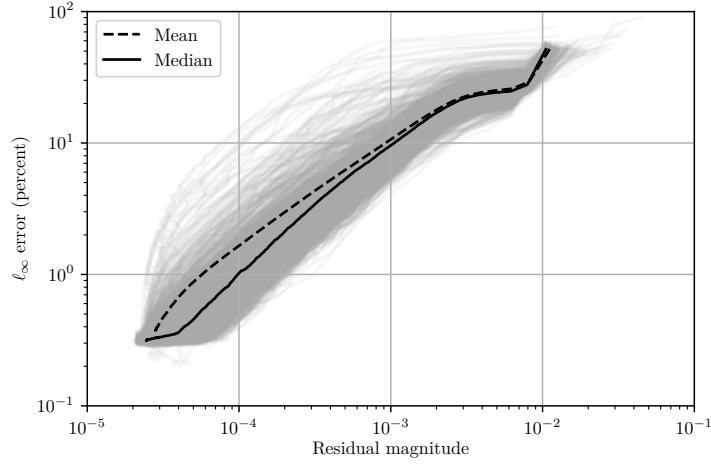
Figure 7: Trajectories showing the evolution of the ℓ_∞ error against the residual loss for the 1000 sound speed distributions in the test set. The ℓ_∞ error is calculated by comparison with k-Wave, and the residual is calculated using a physics loss term. The dashed and solid lines indicate the mean and the median of all traces, respectively.
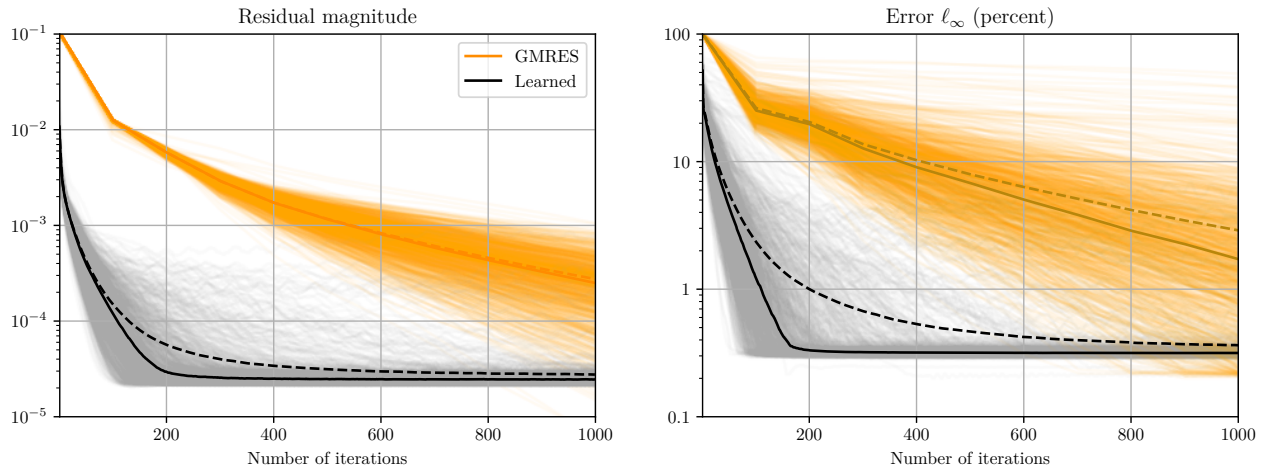

Figure 8: Progression of the optimization problem with the number of iterations for the learned optimizer (black) and the generalized minimal residual (GMRES) method (orange) for the 1000 sound speed distributions in the test set. The dashed and solid lines indicate the mean and the median of the individual traces, respectively. (left) Magnitude of the residual. (right) ℓ_∞ error compared to k-Wave.

(a) Generalized minimal residual (GMRES) method.
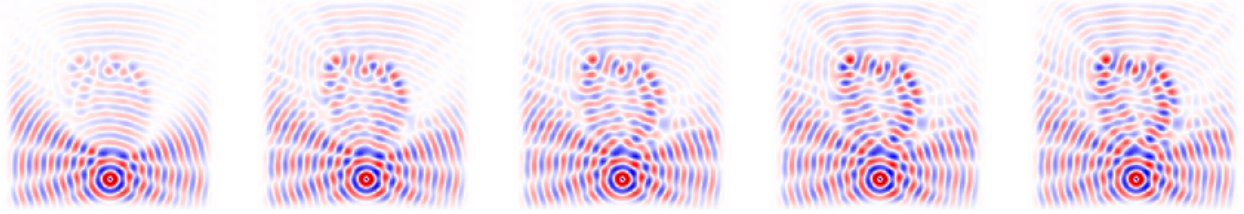
(b) Learned optimizer.

Figure 9: Evolution of the solution after 10, 20, 50, 100, and 250 iterations for GMRES and the learned optimizer for a sound speed distribution from the test set (the real part of the wavefield is shown). The results for GMRES demonstrate the local nature of Krylov iterations based on the Helmholtz operator, which build the solution from the source location outwards. The results for the learned optimizer show a similar dynamic, but the updates cover the global domain much more rapidly.
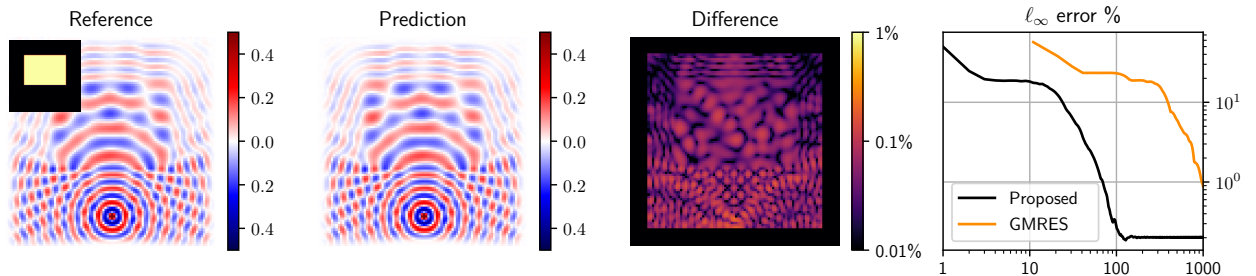

Figure 10: Simulation using a large rectangular heterogeneity (inset top left) which has a shape unlike the speed of sound distributions seen during training. The reference solution is computed using k-Wave and shows very close agreement with the prediction using the learned optimizer (the real part of the wavefield is shown). The relative ℓ_∞ error against the reference solution takes approximately 70 iterations to reach 1% .

to solve previously unseen problems. Similar performance was also observed on a range of other examples analogous to the ones described below.

First, we evaluated the model on a speed of sound distribution containing a rectangular region with a speed of sound of 2, to test the ability of the network to deal with large homogeneous speed of sound regions (recall during training that only idealized skull shapes were used). Figure 10 shows the reference solution calculated using k-Wave, the prediction using the learned optimizer, and the evolution of the error with iteration number. For this example, the learned model reaches a very small final error on the order of 0.2%, and reaches an error below 1% extremely quickly (about 70 iterations).
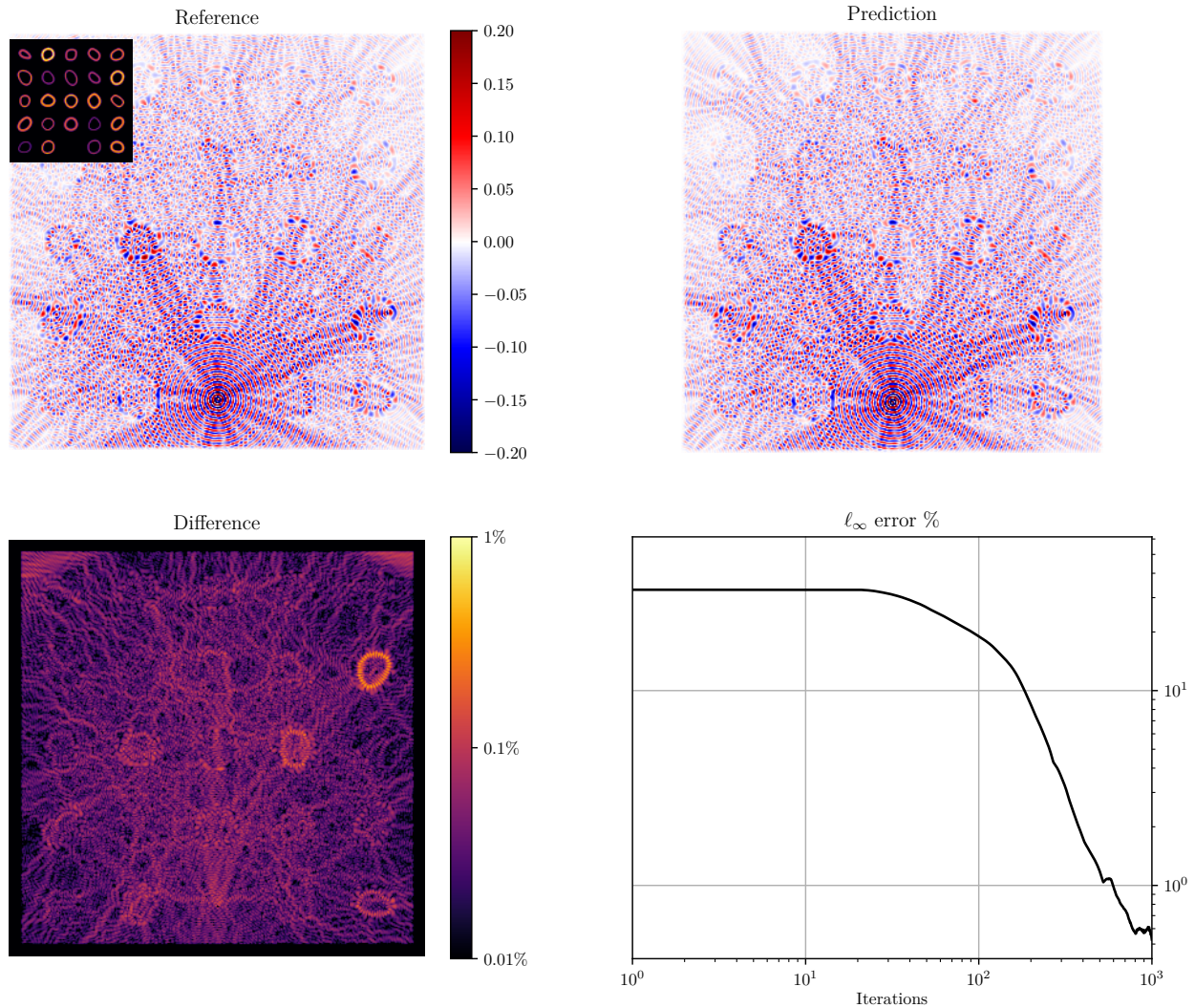
Figure 11: Test on a spatial domain much larger than that seen during training. The sound speed distribution is created by patching together 24 idealized skull distributions from the test set (inset top left). The reference solution is computed using k-Wave and shows very close agreement with the prediction using the learned optimizer, with errors well below 1% (the real part of the wavefield is shown). The relative ℓ_∞ error against the reference solution takes approximately 600 iterations to reach 1%.

Second, we tested the ability of the network to generalize to larger domains. A large speed of sound distribution with 480 × 480 grid points was created by patching together 24 distributions from the test set. Figure 11 shows the reference solution calculated using k-Wave, the prediction using the learned optimizer, and the evolution of the error with iteration number. Despite all the sound speed distributions in the training and validation sets having 96 × 96 grid points, the model is able to generalize to a much larger domain, reaching 1% error within 600 steps. This suggests that the hard task of training on large 3D volumes containing whole skulls for clinical applications can possibly be entirely bypassed by learning the network weights using much simpler and smaller problems, for example, by training using small skull patches.

Finally, while the previous example suggests that the network is able to generalize to different domain sizes, it is unclear how much diversity in the training set is required to ensure that the network still converges to a satisfactory solution with an arbitrary sound speed or source distribution. To test this, we performed a representative simulation of the intracranial wavefield for a transcranial focused ultrasound stimulation experiment [9]. We used a large 512 × 512 speed of sound distribution generated from a transverse CT slice from an adult skull from the Qure.ai CQ500 Dataset [55] converted using the approach outlined in [8]. The source distribution was defined as a focused transducer represented by a 1D arc [56] (recall that the network has only seen single-point sources in a fixed position during training). The transducer aperture diameter and radius of curvature were set to 60 mm, and the source frequency to 490 kHz. We also recall that, due to the linearity of the Helmholtz equation, the wavefield resulting from an arbitrary source distribution can always be decomposed into the sum of wavefields produced by point sources. Therefore it is always possible to run the algorithm in parallel for a set of point-like source maps by decomposing the total source field, with the added benefit of decoupling the effect of each source on the total wavefield.

Figure 12 shows the reference solution calculated using k-Wave, the prediction using the learned optimizer, and the evolution of the error with iteration number. Despite this example being well outside the training set (including a much larger spatial domain, a more complex speed of sound distribution, and a distributed source in an unseen position), the relative ℓ_∞ error compared to k-Wave is very low at 0.8%. This shows that the trained model can be used to solve problems of the scale and complexity needed to make the clinical problem tractable, albeit currently in 2D.

The evolution of the solution shown in Fig. 12 with iteration number is shown in Fig. 13. The solution is constructed from the arc-shaped source outwards. While it takes approximately three thousand iterations for the ℓ_∞ error across the whole domain to reach a minimum, most of the complex structure in the field can be seen after just 200 iterations. Moreover, the position of the focus and the focal pressure (which are not strongly affected by reverberation within the skull for this example) converge extremely quickly, within just 20 iterations. For the purpose of real-time treatment planning (for example, where several candidate positions for the ultrasound transducer may be being evaluated), the ability to generate a reasonable approximation for the acoustic field around the focus in a very short time, which can then be improved by letting the model continue to iterate, is highly desirable. (See Supplementary Material for an example.)

4 Summary and discussion

A lightweight neural network based on a modified UNet is proposed as a fully learned iterative optimizer and used to solve the heterogeneous Helmholtz equation. The network architecture is motivated by multi-scale solvers which utilize multi-scale memory, and Markov decision processes that utilize a belief state to make them fully observable. The network is trained using a physics-based loss function: no explicit knowledge of the true solution is given beyond the implicit consistency with the known source and sound speed.

The training set consists of idealized skulls in a small domain with 96 × 96 grid points, which makes training time manageable (less than one day using 6 GPUs). The learned optimizer shows excellent performance on the test set, and is capable of generalization well outside the training set, including to much larger domains, more complex sound speed distributions, and more complex source distributions.

In the current work, the network is used to solve the lossless Helmholtz equation in 2D. In future, we will look to extend this to 3D, and to include the effects of changes in mass density, acoustic absorption, and potentially the effects of acoustic nonlinearity. The examples given in §3.3 suggest that the model weights could be trained on small 3D patches, which will be essential to make the training tractable.

The neural network has not been constrained in any way and therefore it is hard to state any *a priori* guarantees on its convergence. In general, the convergence of repeatedly applying neural networks with weight-tying is still an open problem (see [57]). However, the network could be constructed in such a way
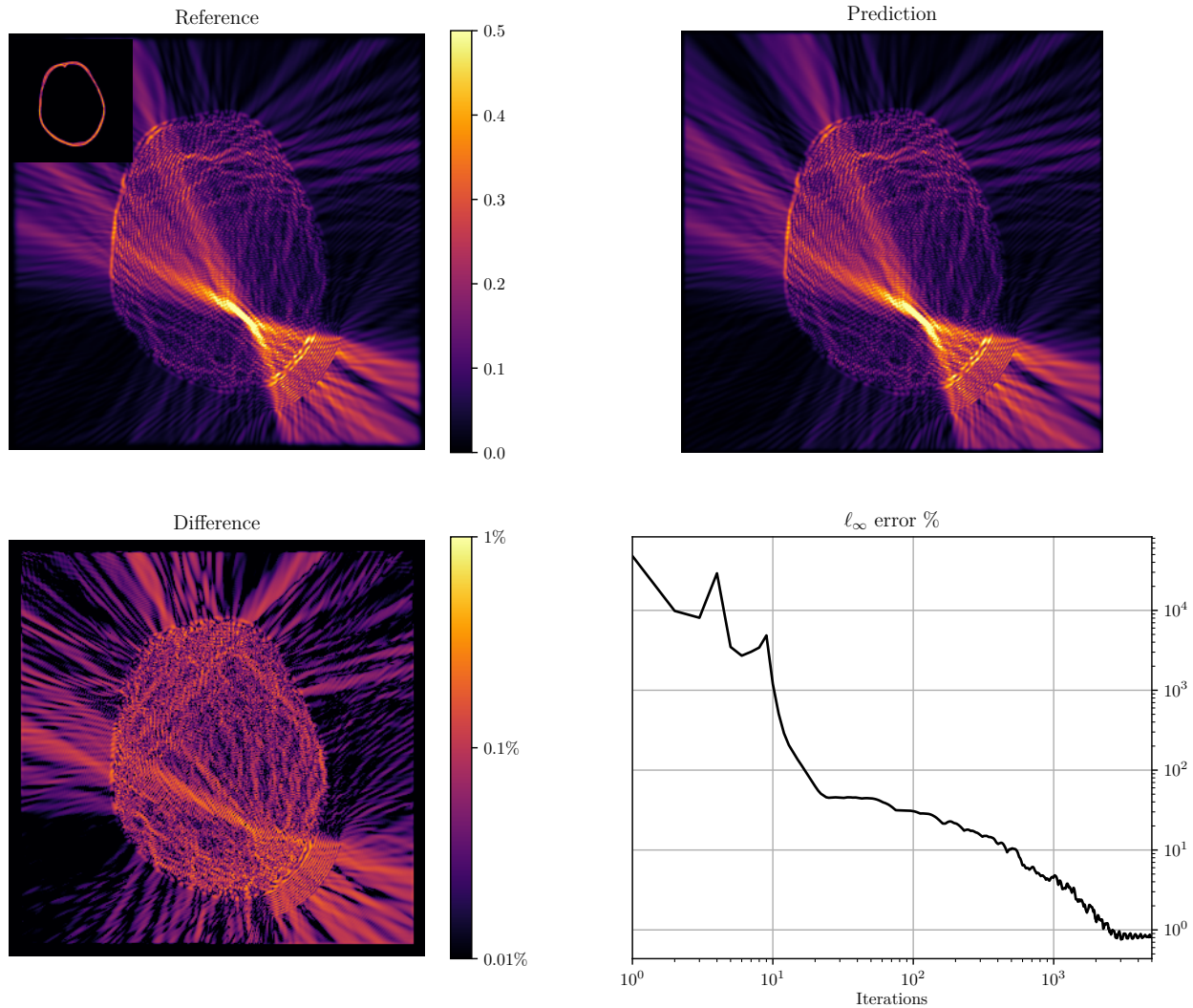
Figure 12: Simulation of a transcranial ultrasound field using a sound speed distribution generated from a clinical x-ray computed tomography (CT) slice (inset top left). The reference solution (top left) is computed using k-Wave and shows very close agreement with the prediction using the learned optimizer (top right), with errors below 1% (bottom left). The absolute value of the complex wavefield is shown. The ℓ_∞ error takes approximately 3000 iterations of the optimizer to reach a minimum (bottom right).

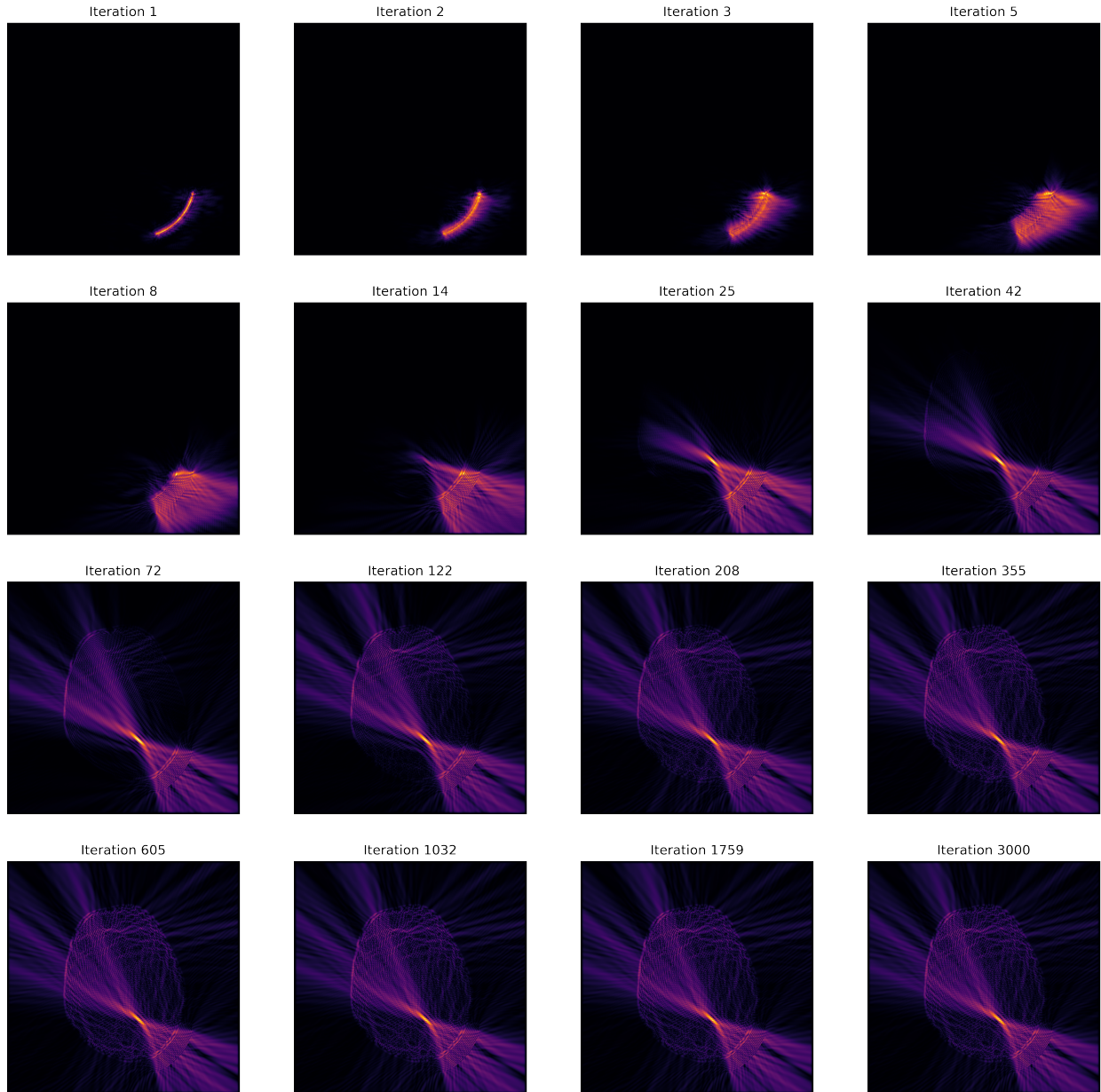Figure 13: Evolution of the wavefield predicted by the learned optimizer for different iteration numbers for the transcranial ultrasound example given in Fig. 12. The position of the focus and the focal pressure converge with 20 iterations, and most of the complex structure in the field can be seen after just 200 iterations. Note, the colormap of each subpanel is normalized by its own maximum.

to ensure at least convergence, for example, by taking optimal step-sizes. Regarding convergence speed, standard iterative solvers often rely on some form of preconditioning to achieve convergence in a small number of iterations. Preconditioners could also be used to enhance the performance of learned solvers, either by applying them at the residual calculation stage or by using them to improve the spectral properties of the physics loss.

One challenge in the formulation that may need addressing is the limited number of unrolling steps used in the training phase. While a replay buffer has proven useful to extend the training horizon well beyond this limit, it may still induce biases or unwanted phenomena such as state representation drift [52]. This could be mitigated by borrowing ideas from the reinforcement learning community, which often deals with large or even infinite horizon tasks. In particular, Q-learning methods [58], such as temporal difference learning, can theoretically take into account the entire sequence of possible future states when providing the loss for a given output, by estimating the future loss. Furthermore, as they are designed to work with externally given reward signals, the loss function on which the network is trained doesn't need to be differentiable. This makes it possible to use more elaborate training strategies, such as imposing a monotonically decreasing residual norm during inference.

Since we are mostly interested in the solution in a restricted region of space, such as the brain for transcranial applications, an interesting extension of the method would be to include a spatial estimate of the uncertainty of the solution at each step [29]. While being useful in and of itself, this may also allow the design of training procedures where the network focuses on solving the problem in the spatial region of interest.

In our experiments, the complexity of the network was kept very low by using a small number of parameters, and this is partially responsible for the good generalization performance. On the one hand, having a small number of parameters reduces the capacity of the network and as a consequence the speed at which a problem is solved. Conversely, having a large number of parameters may overspecialize the network on the distribution of problems encountered in training, which may require a more diverse training dataset to restore its generalization capabilities. Furthermore, the validation loss exhibits large oscillations throughout training, making evaluation and checkpointing of the network parameters subject to a low validation loss a crucial part of the training process. Lastly, the neural network was trained with a fixed frequency relative to the grid spacing (i.e., a fixed number of points per wavelength in the background medium). Therefore it is applicable to all problems as long as the spatial discretization is chosen to satisfy this constraint. For arbitrary frequencies (or, equivalently, grid resolutions) the network currently needs to be retrained or fine-tuned. However this requirement may be relaxed in the future by training the solver on a suitable range of frequencies. While this may reduce the performance of the resulting network, it could also act to regularize the network and allow training larger networks. As part of future work, we will aim to understand and disentangle the effect of dataset diversity, network complexity, training strategies, and the generalization properties of the network.

Lastly, although the neural network solver already provides an advantage over many standard numerical methods by being fully differentiable, the computational performance of the network still needs to be properly assessed. In particular, profiling and optimization of the deployed network, and a comparison to more widespread solving procedures including a measure of their scaling properties for different problem sizes are required. While there may be several small design ideas that can be used to reduce the computational complexity of the network, such as learning the Laplacian operator [59] rather than evaluating it using spectral methods, a fair comparison will require a careful implementation of the method.

Acknowledgments

References

[1] N. McDannold, G. T. Clement, P. Black, F. Jolesz, and K. Hynynen, "Transcranial magnetic resonance imaging–guided focused ultrasound surgery of brain tumors: Initial findings in 3 patients," *Neurosurgery*, vol. 66, no. 2, pp. 323–332, 2010.

[2] W. Legon, T. F. Sato, A. Opitz, J. Mueller, A. Barbour, A. Williams, and W. J. Tyler, "Transcranial focused ultrasound modulates the activity of primary somatosensory cortex in humans," *Nature Neuroscience*, vol. 17, no. 2, pp. 322–329, 2014.

[3] A. Abrahao, Y. Meng, M. Llinas, Y. Huang, C. Hamani, T. Mainprize, I. Aubert, C. Heyn, S. E. Black, K. Hynynen, N. Lipsman, and L. Zinman, "First-in-human trial of blood–brain barrier opening in amyotrophic lateral sclerosis using MR-guided focused ultrasound," *Nature Communications*, vol. 10, no. 1, pp. 1–9, 2019.

[4] K. Hynynen and F. A. Jolesz, "Demonstration of potential noninvasive ultrasound brain therapy through an intact skull," *Ultrasound in Medicine & Biology*, vol. 24, no. 2, pp. 275–283, 1998.

[5] W. S. Chang, H. H. Jung, E. Zadicario, I. Rachmilevitch, T. Tlusty, S. Vitek, and J. W. Chang, "Factors associated with successful magnetic resonance-guided focused ultrasound treatment: Efficiency of acoustic energy delivery through the skull," *Journal of Neurosurgery*, vol. 124, no. 2, pp. 411–416, 2016.

[6] L. Albelda Gimeno, E. Martin, O. Wright, and B. E. Treeby, "Experimental assessment of skull aberration and transmission loss at 270 kHz for focused ultrasound stimulation of the primary visual cortex," in *2019 IEEE International Ultrasonics Symposium (IUS)*, IEEE, 2019.

[7] J. Sun and K. Hynynen, "Focusing of therapeutic ultrasound through a human skull: a numerical study," *The Journal of the Acoustical Society of America*, vol. 104, no. 3, pp. 1705–1715, 1998.

[8] J.-F. Aubry, M. Tanter, M. Pernot, J.-L. Thomas, and M. Fink, "Experimental demonstration of noninvasive transskull adaptive focusing based on prior computed tomography scans," *The Journal of the Acoustical Society of America*, vol. 113, no. 1, pp. 84–93, 2003.

[9] W. Lee, H.-C. Kim, Y. Jung, Y. A. Chung, I.-U. Song, J.-H. Lee, and S.-S. Yoo, "Transcranial focused ultrasound stimulation of human primary visual cortex," *Scientific Reports*, vol. 6, no. 1, pp. 1–12, 2016.

[10] G. Pinton, J.-F. Aubry, M. Fink, and M. Tanter, "Effects of nonlinear ultrasound propagation on high intensity brain therapy," *Medical Physics*, vol. 38, no. 3, pp. 1207–1216, 2011.

[11] A. Pulkkinen, B. Werner, E. Martin, and K. Hynynen, "Numerical simulations of clinical focused ultrasound functional neurosurgery," *Physics in Medicine & Biology*, vol. 59, no. 7, p. 1679, 2014.

[12] S. Almquist, D. L. Parker, and D. A. Christensen, "Rapid full-wave phase aberration correction method for transcranial high-intensity focused ultrasound therapies," *Journal of Therapeutic Ultrasound*, vol. 4, no. 1, 2016.

[13] J. L. Robertson, B. T. Cox, J. Jaros, and B. E. Treeby, "Accurate simulation of transcranial ultrasound propagation for ultrasonic neuromodulation and stimulation," *The Journal of the Acoustical Society of America*, vol. 141, no. 3, pp. 1726–1738, 2017.

[14] N. McDannold, P. J. White, and R. Cosgrove, "Element-wise approach for simulating transcranial MRI-guided focused ultrasound thermal ablation," *Physical Review Research*, vol. 1, no. 3, p. 033205, 2019.

[15] G. T. Clement and K. Hynynen, "A non-invasive method for focusing ultrasound through the human skull," *Physics in Medicine & Biology*, vol. 47, no. 8, p. 1219, 2002.

[16] A. Kyriakou, E. Neufeld, B. Werner, G. Székely, and N. Kuster, "Full-wave acoustic and thermal modeling of transcranial ultrasound propagation and investigation of skull-induced aberration correction techniques: A feasibility study," *Journal of Therapeutic Ultrasound*, vol. 3, no. 1, p. 11, 2015.

[17] K. H. Jin, M. T. McCann, E. Froustey, and M. Unser, "Deep convolutional neural network for inverse problems in imaging," *IEEE Transaction on Image Processing*, vol. 26, no. 9, pp. 4509–4520, 2017.

[18] K. Hammernik, J. Schlemper, C. Qin, J. Duan, R. M. Summers, and D. Rueckert, "$\sum$-net: Systematic evaluation of iterative deep neural networks for fast parallel MR image reconstruction," *arXiv preprint arXiv:1912.09278*, 2019.

[19] P. J. White, G. T. Clement, and K. Hynynen, "Longitudinal and shear mode ultrasound propagation in human skull bone," *Ultrasound in Medicine & Biology*, vol. 32, no. 7, pp. 1085–1096, 2006.

[20] J. Robertson, E. Martin, B. Cox, and B. E. Treeby, "Sensitivity of simulated transcranial ultrasound fields to acoustic medium property maps," *Physics in Medicine & Biology*, vol. 62, no. 7, p. 2559, 2017.

[21] P. B. Rosnitskiy, P. V. Yuldashev, O. A. Sapozhnikov, L. R. Gavrilov, and V. A. Khokhlova, "Simulation of nonlinear trans-skull focusing and formation of shocks in brain using a fully populated ultrasound array with aberration correction," *The Journal of the Acoustical Society of America*, vol. 146, no. 3, pp. 1786–1798, 2019.

[22] B. E. Treeby, J. Jaros, A. P. Rendell, and B. Cox, "Modeling nonlinear ultrasound propagation in heterogeneous media with power law absorption using a k-space pseudospectral method," *The Journal of the Acoustical Society of America*, vol. 131, no. 6, pp. 4324–4336, 2012.

[23] S. Wang, V. Maarten, and J. Xia, "Acoustic inverse scattering via Helmholtz operator factorization and optimization," *Journal of Computational Physics*, vol. 229, no. 22, pp. 8445–8462, 2010.

[24] N. A. Gumerov and R. Duraiswami, "A broadband fast multipole accelerated boundary element method for the three dimensional Helmholtz equation," *The Journal of the Acoustical Society of America*, vol. 125, no. 1, pp. 191–205, 2009.

[25] A. Bermúdez, L. Hervella-Nieto, A. Prieto, R. Rodrı, *et al.*, "An optimal perfectly matched layer with unbounded absorbing function for time-harmonic acoustic scattering problems," *Journal of Computational Physics*, vol. 223, no. 2, pp. 469–488, 2007.

[26] M. Raissi, P. Perdikaris, and G. E. Karniadakis, "Physics informed deep learning (part i): Data-driven solutions of nonlinear partial differential equations," *arXiv preprint arXiv:1711.10561*, 2017.

[27] E. Weinan and B. Yu, "The deep Ritz method: A deep learning-based numerical algorithm for solving variational problems," *Communications in Mathematics and Statistics*, vol. 6, no. 1, pp. 1–12, 2018.

[28] L. Bar and N. Sochen, "Unsupervised deep learning algorithm for PDE-based forward and inverse problems," *arXiv preprint arXiv:1904.05417*, 2019.

[29] Y. Zhu, N. Zabaras, P.-S. Koutsourelakis, and P. Perdikaris, "Physics-constrained deep learning for high-dimensional surrogate modeling and uncertainty quantification without labeled data," *Journal of Computational Physics*, vol. 394, pp. 56–81, 2019.

[30] Y. Saad and M. H. Schultz, "GMRES: A generalized minimal residual algorithm for solving nonsymmetric linear systems," *SIAM Journal on Scientific and Statistical Computing*, vol. 7, pp. 856–869, July 1986.

[31] O. G. Ernst and M. J. Gander, "Why it is difficult to solve helmholtz problems with classical iterative methods," in *Numerical analysis of multiscale problems*, pp. 325–363, Springer, 2012.

[32] Y. A. Erlangga, C. W. Oosterlee, and C. Vuik, "A novel multigrid based preconditioner for heterogeneous Helmholtz problems," *SIAM Journal on Scientific Computing*, vol. 27, pp. 1471–1492, Jan. 2006.

[33] C. W. Oosterlee, C. Vuik, W. A. Mulder, and R.-E. Plessix, "Shifted-Laplacian preconditioners for heterogeneous Helmholtz problems," in *Advanced Computational Methods in Science and Engineering* (B. Koren and K. Vuik, eds.), vol. 71, pp. 21–46, Berlin, Heidelberg: Springer Berlin Heidelberg, 2009. Series Title: Lecture Notes in Computational Science and Engineering.

[34] B. Engquist and L. Ying, "Sweeping preconditioner for the helmholtz equation: hierarchical matrix representation," *Communications on pure and applied mathematics*, vol. 64, no. 5, pp. 697–735, 2011.

[35] H. C. Elman, O. G. Ernst, and D. P. O'leary, "A multigrid method enhanced by Krylov subspace iteration for discrete Helmholtz equations," *SIAM Journal on scientific computing*, vol. 23, no. 4, pp. 1291–1315, 2001.

[36] Y. Chen, B. Dong, and J. Xu, "Meta-mgnet: Meta multigrid networks for solving parameterized partial differential equations," *arXiv preprint arXiv:2010.14088*, 2020.

[37] G. Rizzuti, A. Siahkoohi, and F. J. Herrmann, "Learned iterative solvers for the Helmholtz equation," in *81st EAGE Conference and Exhibition 2019*, vol. 2019, pp. 1–5, European Association of Geoscientists & Engineers, 2019.

[38] O. Ronneberger, P. Fischer, and T. Brox, "U-Net: Convolutional networks for biomedical image segmentation," in *International Conference on Medical image computing and computer-assisted intervention*, pp. 234–241, Springer, 2015.

[39] M. Andrychowicz, M. Denil, S. Gomez, M. W. Hoffman, D. Pfau, T. Schaul, B. Shillingford, and N. De Freitas, "Learning to learn by gradient descent by gradient descent," in *Advances in Neural Information Processing Systems*, pp. 3981–3989, 2016.

[40] P. Putzky and M. Welling, "Recurrent inference machines for solving inverse problems," *arXiv preprint arXiv:1706.04008*, 2017.

[41] B. Zhu, J. Z. Liu, S. F. Cauley, B. R. Rosen, and M. S. Rosen, "Image reconstruction by domain-transform manifold learning," *Nature*, vol. 555, pp. 487–492, Mar. 2018.

[42] K. Li and J. Malik, "Learning to optimize," *arXiv:1606.01885 [cs, math, stat]*, June 2016. arXiv: 1606.01885.

[43] M. Hausknecht and P. Stone, "Deep recurrent Q-learning for partially observable MDPs," *arXiv:1507.06527 [cs]*, Jan. 2017. arXiv: 1507.06527.

[44] Y. Lu, A. Zhong, Q. Li, and B. Dong, "Beyond finite layer neural networks: Bridging deep architectures and numerical differential equations," in *International Conference on Machine Learning*, pp. 3276–3285, PMLR, 2018.

[45] E. Dupont, A. Doucet, and Y. W. Teh, "Augmented neural ODEs," in *Advances in Neural Information Processing Systems*, pp. 3140–3150, 2019.

[46] J. Adler and O. Öktem, "Solving ill-posed inverse problems using iterative deep neural networks," *Inverse Problems*, vol. 33, p. 124007, nov 2017.

[47] K. He, X. Zhang, S. Ren, and J. Sun, "Delving deep into rectifiers: Surpassing human-level performance on imagenet classification," in *Proceedings of the IEEE International Conference on Computer Vision*, pp. 1026–1034, 2015.

[48] J. He and J. Xu, "MgNet: A unified framework of multigrid and convolutional neural network," *Science China Mathematics*, vol. 62, no. 7, pp. 1331–1354, 2019.

[49] F. J. Fry and J. E. Barger, "Acoustical properties of the human skull," *The Journal of the Acoustical Society of America*, vol. 63, no. 5, pp. 1576–1590, 1978.

[50] H. Jaeger, *Tutorial on training recurrent neural networks, covering BPPT, RTRL, EKF and the "echo state network" approach*, vol. 5. GMD Report 159, German National Research Center for Information Technology, 2002.

[51] C. Tallec and Y. Ollivier, "Unbiasing truncated backpropagation through time," *arXiv:1705.08209 [cs]*, May 2017. arXiv: 1705.08209.

[52] S. Kapturowski, G. Ostrovski, J. Quan, R. Munos, and W. Dabney, "Recurrent experience replay in distributed reinforcement learning," in *International Conference on Learning Representations*, 2018.

[53] W. Falcon, "PyTorch Lightning," *GitHub. Note: https://github.com/PyTorchLightning/pytorch-lightning*, vol. 3, 2019.

[54] B. E. Treeby and B. T. Cox, "k-Wave: MATLAB toolbox for the simulation and reconstruction of photoacoustic wave fields," *Journal of Biomedical Optics*, vol. 15, no. 2, p. 021314, 2010.

[55] S. Chilamkurthy, R. Ghosh, S. Tanamala, M. Biviji, N. G. Campeau, V. K. Venugopal, V. Mahajan, P. Rao, and P. Warier, "Development and validation of deep learning algorithms for detection of critical findings in head CT scans," *arXiv preprint arXiv:1803.05854*, 2018.

[56] E. Martin, Y. T. Ling, and B. E. Treeby, "Simulating focused ultrasound transducers using discrete sources on regular Cartesian grids," *IEEE Transactions on Ultrasonics, Ferroelectrics, and Frequency Control*, vol. 63, no. 10, pp. 1535–1542, 2016.

[57] K. Kawaguchi, "A simple deep equilibrium model converges to global optima with weight tying," 2021.

[58] R. S. Sutton and A. G. Barto, *Reinforcement learning: an introduction.* Adaptive computation and machine learning, Cambridge, Mass: MIT Press, 1998.

[59] Z. Long, Y. Lu, X. Ma, and B. Dong, "PDE-Net: Learning PDEs from data," in *International Conference on Machine Learning*, pp. 3208–3216, 2018.

[60] J. P. Boyd, *Chebyshev and Fourier spectral methods.* Dover, Mineola, 2001.

[61] X. Glorot and Y. Bengio, "Understanding the difficulty of training deep feedforward neural networks," in *Proceedings of the thirteenth international conference on artificial intelligence and statistics*, pp. 249–256, JMLR Workshop and Conference Proceedings, 2010.

A Discrete solution of the forward operator

The calculation of the residual in Eq. (8) and the loss function in Eq. (11) require the discretization of the heterogeneous Helmholtz equation in Eq. (1). To allow the Sommerfeld radiation condition at infinity to be approximately satisfied while cropping the domain to a finite size $\Omega \subset \mathbb{R}^2$, a perfectly matched layer or PML is used as defined in [25]. Assuming the original bounded domain is rectangular with size $2L_x$ by $2L_y$, the domain is extended to a size of $2(L_x + \Delta L)$ by $2(L_y + \Delta L)$, where ΔL is the thickness of the PML on each side of the domain (see Fig. 1). The derivative operators are then transformed to introduce absorption within the extended part of the domain:

$$\frac{\partial}{\partial \eta} \to \frac{1}{\gamma_\eta} \frac{\partial}{\partial \eta} \ , \tag{13}$$

where $\eta = x, y$ and

$$\gamma_\eta = \begin{cases} 1, & \text{if } |\eta| < L_\eta \\ 1 + \dfrac{j}{\omega} \sigma(x), & \text{if } L_\eta \leq |\eta| < L_\eta + \Delta L \ . \end{cases} \tag{14}$$

The absorption profile σ grows quadratically within the perfectly matched layer according to [25]

$$\sigma(\eta) = \sigma_{\max} \left(1 - \frac{\eta}{L} \right)^2 . \tag{15}$$

The Laplacian including the PML then becomes

$$\hat{\nabla}^2 = \frac{1}{\gamma_x} \frac{\partial}{\partial x} \left(\frac{1}{\gamma_x} \frac{\partial}{\partial x} \right) + \frac{1}{\gamma_y} \frac{\partial}{\partial y} \left(\frac{1}{\gamma_y} \frac{\partial}{\partial y} \right) = \sum_{\eta=x,y} \left[\frac{1}{\gamma_\eta^2} \frac{\partial^2}{\partial \eta^2} - \frac{\gamma_\eta'}{\gamma_\eta^3} \frac{\partial}{\partial \eta} \right] \ , \tag{16}$$

where $\gamma_\eta' = \frac{\partial}{\partial \eta} \gamma_\eta$, which is computed analytically. To discretize Eq. (16), the Fourier collocation spectral method is used [60], where

$$\frac{d}{d\eta} f(\eta) = \mathcal{F}_\eta^{-1} \left\{ ik_\eta \mathcal{F}_\eta \left\{ f(\eta) \right\} \right\} \ . \tag{17}$$

Here k_η are the wavenumbers in the η-direction (see e.g., [60]), and $\mathcal{F}$ and $\mathcal{F}^{-1}$ represent the forward and inverse Fourier transform, in this case computed using the fast Fourier transform (FFT). Finally, this gives

$$\hat{\nabla}^2 f(x,y) = \sum_{\eta=x,y} \left[\frac{1}{\gamma_\eta^2} \mathcal{F}_\eta^{-1} \left\{ (-k_\eta^2) \mathcal{F}_\eta \left\{ f(x,y) \right\} \right\} - \frac{\gamma_\eta'}{\gamma_\eta^3} \mathcal{F}_\eta^{-1} \left\{ (ik_\eta) \mathcal{F}_\eta \left\{ f(x,y) \right\} \right\} \right] \ . \tag{18}$$

For a numerical calculations presented in the current work, the PML size ΔL is set to 8, and the absorption parameter $\sigma_{\max}$ is set to 2.

B Training algorithm

The algorithm used for training is given in Algorithm. 1

Algorithm 1: Training procedure

Data: Train set $\mathcal{C}$, Validation set $\mathcal{V}$, source ρ, buffer size N_s, maximum iteration T, unrolling steps t, batch size N_b

```
/* Initialize replay buffer                                             */
```
1 for $n \in [1, \ldots, N_s]$ **do**
2 $k \sim \mathcal{U}(0, T-1)$
3 $c \sim \mathcal{C}$
4   Initialize u_k, h_k with zeros
5   Store (c, u_k, h_k) in the buffer B

```
/* Training loop                                                        */
```
6 $\hat{V} \leftarrow \infty$ `// Initialize best validation loss to infinity`
7 for $epoch \in \{1, \ldots, N_{epochs}\}$ **do**
```
    /* Train network for a single epoch                                 */
```
8   **for** $step \in \{1, \ldots, N_{steps}\}$ **do**
9   Sample a batch of N_b triplets (c, u_k, h_k) from B
```
        /* For each triplet                                             */
```
10   **for** $n \in \{1, \ldots, N_b\}$ **do**
```
            /* Unroll t steps                                           */
```
11   **for** $i \in \{k+1, \ldots, k+t\}$ **do**
12 $e_{i-1} \leftarrow A(c)u_{i-1} - \rho$
13 $u_i \leftarrow u_{i-1} + f_\theta(u_{i-1}, e_{i-1}, h_{i-1})$
14 $L_n = \sum_i L_{i,n}$ `// Estimate the loss over t steps`
15
```
            /* Update replay buffer                                     */
```
16 $i \sim \mathcal{U}(k+1, k+t)$ `// Sample one of the t iterations`
17   **if** $i < T$ **then**
18   Store (c, u_i, h_i) in the buffer B in place of its original triplet
19   **else**
20 $c \sim \mathcal{C}$
21   Initialize u_0, h_0 with zeros
22   Store (c, u_0, h_0) in the buffer B in place of its original triplet
23 $R \leftarrow \frac{1}{N_b} \sum_n L_n$ `// Estimate the loss over t steps`
24 $\theta \leftarrow SGD(\theta, \nabla_\theta R)$ `// Update parameters via gradient descent`
```
    /* Evaluate the network on the validation set every 10 epochs        */
```
25   **if** $mod(epoch, 10) = 0$ **then**
26 $L_{val} \leftarrow 0$
27   **for** $every\ c \in V$ **do**
28   Initialize u_0, h_0 with zeros
29   Initialize the source map ρ with a point source at a random position
30   **for** $i \in \{1, \ldots, T\}$ **do**
31 $e_{i-1} \leftarrow A(c)u_{i-1} - \rho$
32 $u_i \leftarrow u_{i-1} + f_\theta(u_{i-1}, e_{i-1}, h_{i-1})$
33 $L \leftarrow L_{T,n}$ `// Estimate the loss at the last iteration`
34 $L_{val} \leftarrow L_{val} + L$ `// Add to the total loss over the validation set`
35   **if** $L_{val} < \hat{V}$ **then**
36 $\hat{V} \leftarrow L_{val}$
37   Save the current network parameters θ

C Model and training hyperparameters

A summary of the hyperparameters used in the network and for training is given in Tables 1 and 2.   All the convolutional layers were initialized using the Xavier method with gain 0.02 [61].

Table 1: Model hyperparameters

Parameter	Value
activation function	PReLU
depth	4
channels per layer	8
channels per state	2

Table 2: Training hyperparameters

Parameter	Value
batch size	32
buffer size	600
gradient clipping	1
learning rate	10^{-4}
weight decay	10^{-6}
window size for TBPTT	10